

Customer-Centric Bank Strategies Using Unsupervised Machine Learning

A1 - Team 11

Members:

Divyesh Sai Annavarapu

Junhan Chen

Nimisha Agarwal

Rashi Jaiswal

Date: 3 March 2025

Proposal

Banks need to understand customers beyond basic demographics. Traditional segmentation methods based on age, income, or job type overlook real financial behavior. Some customers prioritize savings, others take frequent loans, and some engage more digitally. To move beyond such assumptions, our project aims to apply unsupervised machine learning to segment customers based on actual banking patterns.

Our study focuses on three key objectives:

1. **Effective Customer Segmentation** – Categorizing customers by financial habits and engagement rather than broad demographics.
2. **Targeted Marketing Strategies** – Using data-backed segments to personalize financial products and improve campaign success.
3. **Optimized Resource Allocation** – Helping banks focus efforts efficiently, reducing marketing waste, and enhancing customer retention.

By leveraging clustering techniques and association rule mining, banks can offer tailored financial solutions that align with real customer behaviors, improving both customer experience and business growth.

Exploratory Data Analysis (EDA)

Our dataset of **45,211 customer records** from a Portuguese bank's marketing campaigns (May 2008–November 2010) offers insights into customer interactions with financial products. Collected through phone outreach, it captures demographics, financial behavior, and past engagement.

Most customers are **30–50 years old**, making them prime candidates for loans, savings, and investments. The largest groups work in **blue-collar, management, or technician roles**, suggesting tailored financial services for these professions. Since most are **married**, there's potential for family-focused banking products. **Education impacts engagement**, with tertiary-educated customers responding slightly better, likely due to greater financial awareness.

Financial behavior varies widely. The average balance is €1,362, but most have much **lower balances**, and 16% are in negative balance. Only **6.3%** hold **high balances**, requiring targeted premium banking offers. Customers without personal loans tend to have higher balances, while those with loans have lower ones, suggesting **financial stability reduces reliance on credit**. Housing loans don't strongly correlate with account balances, indicating factors like job stability and credit history play a bigger role in mortgage approvals.

The marketing campaign had a **low success rate**. Retired individuals responded best, likely due to financial stability and time for long-term planning. Entrepreneurs, self-employed individuals, and students engaged the least, possibly due to financial uncertainty or preference for flexible financial products. Blue-collar workers and management professionals were engaged but had low conversions, suggesting the

messaging wasn't compelling. Secondary-educated customers participated most but subscribed the least, highlighting the need for better-targeted marketing.

These insights highlight the importance of data-driven customer segmentation over basic demographics, enabling banks to refine marketing and product recommendations for better engagement and conversions.

Preprocessing of dataset

We cleaned and structured the dataset to focus on meaningful customer patterns. Columns like "**day**," "**month**," "**pday**," "**y**," and "**duration**" were dropped since they didn't add value for clustering. Key categorical features were transformed using **TransactionEncoder** for association rule mining. To improve clustering, we **standardized** balance and campaign interactions and **removed redundant or highly correlated variables**. These steps ensured the dataset was optimized for accurate segmentation and meaningful insights.

Analysis Plan

To build meaningful customer segments, we applied multiple unsupervised learning techniques, refining our approach iteratively for better segmentation accuracy. Our goal was to classify customers based on financial behavior, previous interactions, and loan history while also identifying outliers and uncovering hidden patterns using association rule mining.

Hierarchical Clustering (Ward's Method)

We began with Hierarchical Clustering using Ward's linkage, which minimizes intra-cluster variance, ensuring well-separated groups. While it provided a **better structure** than K-Means for mixed data, it struggled with scalability, making it **slow** for large datasets. To optimize performance, we used **subsampling** and fine-tuned cluster thresholds using **fcluster**, addressing imbalanced cluster sizes.

K-Prototypes Clustering

To handle mixed data more effectively, we implemented **K-Prototypes**, which combines K-Means for numerical data and K-Modes for categorical features. Choosing the right **initialization method** was key. Huang's method produced inconsistent clusters, so we switched to **Cao's method**, which ensured stability. We also fine-tuned **max_iter**, testing values of 10, 30, and 60, ultimately selecting **30 iterations** as the best balance between efficiency and accuracy.

DBSCAN for Outlier Detection

For detecting financial outliers, we used DBSCAN, which identifies dense clusters while marking sparse data points as noise. Since it is highly sensitive to **eps** (epsilon) and **min_samples**, initial results flagged **too many false positives**. We refined the approach by restricting DBSCAN to **age and balance**, leading to better anomaly detection and highlighting both **high-value customers** and **at-risk individuals**.

Feature Engineering: Total Interactions

To enhance **segmentation**, we created a **Total Interactions** feature by combining ‘campaign’ and ‘previous contacts’. This provided a clearer measure of engagement rather than analyzing these features separately. We applied **feature scaling** to ensure uniformity across variables, improving clustering performance by differentiating **low vs. high-engagement customers**.

Association Rule Mining (Apriori Algorithm)

To extract patterns between categorical attributes, we used Association Rule Mining (ARM) with the Apriori Algorithm. Since ARM requires categorical data in a transactional format, we used **TransactionEncoder** for encoding. Finding the right **support and confidence thresholds** was key, too high yielded few insights; too low introduced noise. Through multiple iterations, we refined these thresholds to generate strong, interpretable rules.

Parameter Tuning and Adjustments

Fine-tuning parameters played a critical role across methods. In Hierarchical Clustering, Ward’s method **minimized variance**; in K-Prototypes, Cao’s initialization **improved stability**; DBSCAN required careful **eps and min_samples selection**; and ARM needed iterative **threshold adjustments**. Each refinement improved segmentation accuracy, ensuring our model effectively categorized customers based on real financial behaviors.

Challenges, Dead Ends & Adjustments

K-Means struggled with mixed data, leading to weak clusters. We tested **Hierarchical Clustering** (Ward’s method), which improved separation but required validation. It was also too computationally expensive, so we used subsampling and adjusted cluster thresholds with fcluster to maintain balance without excessive overhead. **K-Prototypes proved best**, efficiently handling both numerical and categorical data, with Cao’s initialization ensuring stability.

DBSCAN initially misclassified too many outliers due to mixed data sensitivity. Restricting it to numerical features (balance, age) and fine-tuning eps and min_samples significantly improved accuracy.

Early ARM **results** were **weak** due to low support and confidence values. Optimizing thresholds and using TransactionEncoder refined categorical representations, revealing insights like secondary education’s influence on campaign engagement.

Feature engineering improved clustering accuracy. Introducing Total Interactions (campaign + previous contacts) clarified engagement, and applying feature scaling resolved distortions from imbalanced numerical variables.

Insights & Findings

Our analysis identified three key customer **segments** based on **financial behavior**: **Young professionals** have lower balances, minimal financial commitments, and prefer digital banking. **Middle-aged homeowners** show financial stability, engage moderately, and are ideal for mortgage refinancing and insurance products. **High-income investors** hold large balances and prefer wealth management over term deposits.

Financial history analysis showed that **account balance** alone **doesn't predict term deposit interest**. High-balance customers often prefer other investments. Loan status also had no strong correlation with subscriptions, indicating that **engagement history matters** more. Customers with past positive interactions were more likely to engage again, emphasizing the need for targeted re-engagement rather than broad financial status-based marketing.

Demographics played a role in engagement. **Secondary-educated** customers participated more but converted less, while **tertiary-educated** individuals responded better, likely due to financial literacy. **Married customers**, particularly those with secondary education, were less likely to subscribe, suggesting competing financial priorities. Age had little impact on subscriptions, reinforcing the importance of behavior-driven targeting.

Marketing effectiveness depended on **communication methods**. Cellular outreach increased subscriptions, especially for secondary-educated customers, but was not enough alone. **Follow-ups improved conversions**, and multi-channel strategies integrating email, app notifications, and digital marketing proved essential.

Campaign analysis showed that **past success strongly predicted future engagement**, making targeted re-engagement key. However, campaign duration had no significant impact, reinforcing that effective targeting and messaging matter more than outreach length.

To optimize banking strategies, financial products should be tailored to customer behavior. Young professionals benefit from **micro-loans and automated savings**, homeowners from **mortgage refinancing and insurance**, and high-income investors from **premium investment services**.

Prioritizing past-engaged customers in marketing improves efficiency, while **financial literacy** programs and better data collection can enhance engagement among underrepresented groups.

Loan repayment behavior, not just loan status, should **guide deposit marketing**, and structured savings programs should be designed for financially vulnerable customers.

Short, targeted campaigns based on engagement history can maximize conversions, shifting banks from broad demographic-based marketing to precise, behavior-driven strategies.

Appendix

Contribution:

Team Member	Tasks	Challenges
Chen	Worked on preprocessing , applied K-Means clustering and PCA for dimensionality reduction . Tested K-Means on customer segmentation.	K-Means struggled with categorical data , leading to poor cluster separation. PCA didn't provide clear dimensionality reduction as variance was spread across too many components.
Rashi	Handled preprocessing , applied DBSCAN for anomaly detection along with PCA to improve feature selection	Need to explore DBSCAN with mixed data types with a focus on categorical features. Needed to refine the approach to detect meaningful outliers.
Div	Applied Hierarchical Clustering (Ward's Method) for customer segmentation. Worked on preprocessing to optimize feature selection. Drafted initial slides submission.	Determining the optimal number of clusters was challenging. Hierarchical Clustering worked better than K-Means but still required refinement required
Nimisha	Conducted Exploratory Data Analysis (EDA) to understand data distributions and outliers. Applied Association Rule Mining (ARM) using the Apriori Algorithm . Drafted, compiled and finalised project report and initial slide submission.	Categorical variables made clustering difficult , requiring extensive preprocessing. Adjusting support and confidence thresholds in ARM was necessary to extract meaningful rules

GitHub Project

- Repository Link: <https://github.com/Chen-wsnd/BA-820-team-11>

AI Declaration

Some visualizations and statistical methods were refined through ChatGPT. Additionally, Gemini (Google, 2025) within the Colab environment was used to assist with error resolution and

additional support. Grammarly (Grammarly, 2025) was utilized to refine language, enhance clarity, and ensure a professional tone in the analysis. All AI-generated content was critically reviewed and modified to align with the project's objectives.

References:

1. Cognizant. (n.d.). *Customer segmentation in banking*. Retrieved February 19, 2025, from <https://www.cognizant.com/us/en/glossary/customer-segmentation-banking>
2. Lumin Digital. (2022, May 11). *Why customer segmentation in banking is crucial for marketing and service offerings*. Retrieved February 19, 2025, from <https://lumindigital.com/insights/why-customer-segmentation-in-banking-is-crucial-for-marketing-and-service-offerings/>
3. **European Commission**. (n.d.). *[Title of the document, if available]*. European Commission. https://economy-finance.ec.europa.eu/document/download/b3c3bddc-ba2b-4ecd-8274-bf851788ddb5_en?filename=ip285_en.pdf
4. **Global Citizen Solutions**. (n.d.). *Banks in Portugal*. Global Citizen Solutions. <https://www.globalcitizensolutions.com/banks-in-portugal/>
5. **S&P Global**. (n.d.). *[Title of the document, if available]*. S&P Global. https://www.spglobal.com/_assets/documents/ratings/research/101611992.pdf
6. ChatGPT. (2025). Categorizing customer balances into meaningful groups for analysis [Large language model output]. OpenAI. <https://chatgpt.com/share/67b6403d-b380-8002-90b1-317d8b7ba6de>
7. OpenAI. (2025). ChatGPT (Feb 18 version) [Large language model]. OpenAI. <https://chat.openai.com>
8. Google. (2025). Gemini [Large language model]. Google AI. <https://ai.google>
9. Grammarly. (2025). Grammarly [AI writing assistant]. Grammarly Inc. <https://www.grammarly.com>



How can I create balance categories (e.g., Negative, Low, Medium, High) based on balance amounts in a DataFrame?

To create balance categories (e.g., "Negative," "Low," "Medium," "High") based on balance amounts in a pandas DataFrame, you can use the `pd.cut()` function to define fixed bins or `pd.qcut()` to create quantile-based bins.

Method 1: Using `pd.cut()` with Fixed Bins

This method allows you to define explicit balance ranges.

```
python
```

[Copy](#) [Edit](#)

```
import pandas as pd
```

Ask ChatGPT anything

Rashi's Gen AI declaration :

Gen AI declaration for DBSCAN with PCA for outlier detection

1. Looked up DBSCAN to understand what this methodology is about.
2. Introduction to DBSCAN method

for dbscan

DBSCAN: Density-Based Clustering Essentials ~...
datanovia

DBSCAN - Wikipedia
en.wikipedia

DBSCAN Clustering Algorithm Demystified ~...
builtin

+3 sources

Perplexity

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular clustering algorithm in machine learning ¹ ². It groups closely packed data points and identifies outliers without requiring a predefined number of clusters ³ ⁴.

Key Parameters

DBSCAN requires two important parameters:

1. **epsilon (eps)**: The maximum distance between two points to be considered neighbors ¹ ⁴.
2. **minimum points (MinPts)**: The minimum number of points required to form a dense region ¹ ⁴.

How DBSCAN Works

1. Select a starting point and examine its neighborhood within the eps distance ⁴.
2. If there are at least MinPts points within eps distance, mark it as a core point and form a new cluster ⁴.
3. Expand the cluster by adding neighbors of core points recursively ⁴.
4. Repeat the process for all unvisited points ⁴.
5. Points not belonging to any cluster are labeled as noise ⁴.

Advantages

- Finds arbitrarily shaped clusters ².
- Robust to outliers ².
- Does not require specifying the number of clusters beforehand ² ³.

3. This was to know which variable will require what kind of encoding

for 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'poutcome',
for which ones i use one hot encoding, label
encoding, binary encoding

Encoding categorical data for Power BI: Label...
endjin

What Is One Hot Encoding and How to...
datacamp

Label Encoding vs. One Hot Encoding: What's th...
statology

+5 sources

Perplexity

Based on the nature of the variables you've listed, here's how you should encode them:

One-hot encoding:

- Job
- Marital
- Contact

These variables have no inherent order or ranking between categories, making one-hot encoding suitable to prevent false relationships [1](#) [2](#).

Label encoding:

- Education

Education levels typically have an ordinal relationship, so label encoding can preserve this order [3](#).

Binary encoding:

- Default
- Housing
- Loan

These variables likely have only two categories (e.g., yes/no), making them suitable for binary encoding [3](#) [7](#).

Remember that the order of operations can impact model performance. One-hot encoding

4. This is to see what steps were required for conducting DBSCAN, understanding key parameters to be included.

Here's how you can proceed systematically:

1. Preprocessing the Data

- **Handle Missing Values:** Check if there are missing values and impute/drop them accordingly.
- **Encoding Categorical Features:** Convert categorical variables into numerical representations (One-Hot Encoding or Label Encoding).
- **Feature Scaling:** Normalize numerical features using `MinMaxScaler` or `StandardScaler` to ensure DBSCAN's distance-based approach works effectively.

2. Applying DBSCAN

- **Choosing Epsilon (ϵ) and MinPts:**
 - Use a **k-distance plot** to find the optimal ϵ (eps) value.
 - Set **MinPts** as `2 * number_of_features` (rule of thumb).

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=ε, min_samples=MinPts)
clusters = dbscan.fit_predict(X_scaled)
```

[Copy](#) [Edit](#)

- **Identifying Outliers:** DBSCAN labels outliers as `-1`. Extract these records for further analysis.

3. Evaluating the Clusters

- **Visualization:** Use **t-SNE** or **PCA** to reduce dimensions and visualize clusters.
- **Cluster Characteristics:**
 - Compute **mean**, **median**, **standard deviation** for each cluster.
 - Compare **categorical feature distributions** across clusters.


4. Analyzing Outstanding Individuals (Anomalies)

- Investigate customers classified as **outliers**.




5. This is was to know how to conduct this in Python. Although the code below has addressed some additional points, I have not included all the codes found below in my coding exercise for this project. I did use to detect the number of outliers.

how to do dbscan for outlier detection in python

DBSCAN for Outlier Detection in Python ~...
 pieriantraining

Clustering Algorithms for Outlier Detection in...
 github

DBSCAN detects outliers based on what ? and...
 stackoverflow

   
+4 sources

Perplexity

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an effective algorithm for outlier detection in Python. Here's how to implement DBSCAN for outlier detection:

1. Import necessary libraries:

```
python
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

2. Prepare your data:

```
python
# Generate or load your dataset
X = np.random.randn(100, 2)
X = np.vstack((X, [[5, 5], [6, 6], [-5, -5]])) # Add outliers

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

3. Apply DBSCAN:

python



```
# Define DBSCAN parameters
eps = 0.5
min_samples = 5

# Create and fit the DBSCAN model
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
labels = dbscan.fit_predict(X_scaled)
```

4. Identify outliers:

python



```
# Points labeled as -1 are considered outliers
outliers = X_scaled[labels == -1]
inliers = X_scaled[labels != -1]

print("Number of outliers:", len(outliers))
```

5. Visualize results:

5. Visualize results:

python



```
plt.scatter(inliers[:, 0], inliers[:, 1], c='blue', label='Inliers')
plt.scatter(outliers[:, 0], outliers[:, 1], c='red', label='Outliers')
plt.title("DBSCAN Outlier Detection")
plt.legend()
plt.show()
```

DBSCAN detects outliers based on the density of data points in a neighborhood ³. Points that do not belong to any cluster (labeled as -1) are considered outliers ¹ ⁶. The key parameters for DBSCAN are:

- ``eps``: The maximum distance between two samples for them to be considered as part of the same neighborhood.
- ``min_samples``: The minimum number of samples in a neighborhood for a point to be considered as a core point.

To optimize DBSCAN for outlier detection:

1. Normalize your data to ensure features are on the same scale ².
2. Experiment with different ``eps`` and ``min_samples`` values to find the best combination for your dataset ².
3. Use domain knowledge to guide parameter selection ².

can it handle categorical variables

DBSCAN Clustering with Numerical and...

 stackoverflow

Unsupervised clustering with mixed categorical...

 tomasbeuzen

Clustering with categorical and numeric...

 stats.stackexchange

+3 sources

Perplexity

DBSCAN cannot directly handle categorical variables ¹ ⁴. The algorithm is based on Euclidean distances, which are not meaningful for categorical data ¹. To use DBSCAN with datasets containing categorical variables, you have a few options:

1. Transform the categorical data: Convert categorical variables into a numerical format that allows for meaningful distance calculations. This could involve techniques like one-hot encoding ².
2. Use a different distance metric: Employ a distance measure that can handle mixed data types, such as Gower's distance ³ ⁴.
3. Apply dimensionality reduction: Use techniques like Factor Analysis of Mixed Data (FAMD) to convert mixed data into continuous features before clustering ².
4. Consider alternative algorithms: For datasets with primarily categorical data, you might want to explore algorithms specifically designed for categorical clustering, such as k-modes or k-prototypes ² ⁵.

It's important to note that while these approaches can allow you to use DBSCAN with datasets containing categorical variables, they may alter the original data structure or require careful interpretation of results.

2. Handle Categorical Features:

The categorical features (job, marital, education, default, housing, loan, contact, month, outcome, y) cannot be directly used in DBSCAN without preprocessing. You have a few options:

- a) Exclude them initially: Start with just the numerical features for a basic clustering.
- b) Encode them: Use techniques like one-hot encoding or label encoding to convert categories to numerical values. However, this can significantly increase the dimensionality of your data and may not always be meaningful for distance-based algorithms like DBSCAN.
- c) Create derived numerical features: For example, you could create a "risk score" based on combinations of categorical variables.


3. Feature Selection/Reduction:

Consider using techniques like Principal Component Analysis (PCA) to reduce the dimensionality of your data, especially if you decide to include encoded categorical features.

4. Scaling:

It's crucial to scale your numerical features before applying DBSCAN. Use techniques like StandardScaler or MinMaxScaler to ensure all features are on a similar scale.

5. Approach:

- Start with just the 7 numerical features, properly scaled.
- If results are not satisfactory, carefully  incorporate some of the most relevant categorical features after encoding.

